Data Processing and Visualization Toolbox (DPV)

User Manual

Table of Content

Chapter 1 Introduction

1.1 What is Data Processing and Visualization Toolbox (DPV)

DPV Toolbox is extracted from the Neurophysiology Toolbox (NPT). It is a Matlab toolbox with object-oriented programming framework. This toolbox is able to process large data sets, allows developing new analysis algorithms for user-defined objects, and provides Graphical User Interface to inspect the analysis results.

This toolbox is suitable for the data sets with hierarchy/level. The names of the items/directories in the same level should follow certain pattern.

1.2 Structure of DPV

This toolbox consists of the following items:

- Two parent classes, @nptdata and @nptgroup;
- Assistant class @event
- Child class template @dirfiles
- Category gui
- Category miscellaneous

User may create their desired objects using the template @dirfiles, and then use parent class functions to process the data sets to obtain the object in each directory.

Chapter 2 Setup

First of all, make sure Matlab has been installed on the computer. Then do the following steps.

Step 1: Get a copy of the DPV.

Step 2: Create directory where you would like to store DPV and save the toolbox in that directory.

Step 3: Add path.

1. Start MATLAB
2. Change directories to the DPV directory:
   ```
   >> cd ~/matlab/DPV
   ```
3. Add the sub-directories to your path:
   ```
   >> nptAddPath
   ```
4. Go to File -> Set Path;
5. In the dialog box, click "Save" and "Close";

Step 4: Set up the configuration file.

1. In the "Current Directory" Window, select the directory contains the toolbox
2. Copy the file named ConfigurationTemplate.txt
3. In "Command Window", type
   `>> cd(prefdir)`
   The "Current Directory" Window now shows the preference directory.
4. Paste ConfigurationTemplate.txt in the directory, and rename it to be Configuration.txt.
5. In "Command Window", run the function *ModifyConfig.m* to modify the configuration file. You may type the following command in "Command Window" to get help on how to use this function (see Section 3.2).
   `>> help modifyConfig`

Chapter 3 Configuration File

3.1 Contents and Location

The configuration file created in DPV Toolbox is used to save the hierarchy (level) information:

1) nptDataDir – the root directory used to store the data sets;
2) Level names – Contains the name of each level in data hierarchy;
3) Abbreviations of level names – the abbreviations of level names used when creating directories;
4) String pattern of level names – the way to name the directory in each level. This information is useful for the program to figure out the individual directories when in a combinational directory (e.g.: the string pattern of directories in Cluster level is 'cluster01s');
5) Equivalent level names – in case one level has some equivalent levels.

The configuration file is saved in the preference directory, returned by the command 'prefdir'. If more than one users share a single copy of Matlab in one computer, these users can still keep individual preferences.

Configuration file needs to be set up by the individual users respectively. A template configuration file is provided in DPV, named 'ConfigurationTemplate.txt'. This file contains the default data hierarchy information. If user does not set up his own configuration file, the default information will be read during processing. The default values are listed below.

nptDataDir – /var/automount/opt/data/cat;
Level names – Cluster, Group, Session, Site, Day, Days;
Abbreviations of level names – c -> Cluster; g -> Group; n -> Session; s -> Site;

String pattern of level names – cluster00s -> Cluster, group0000 -> Group, session00 -> Session, site00 -> Site.
Equivalent level names – Group/Sort/HighPass/Eye/EyeFilt/Lfp.

3.2 Modifying Configuration File

The function *ConfigModify.m* enables the users to modify any number of the five components at one time.

Examples:

ModifyConfig('nptDataDir', '/var/automount/opt/data/cat');

'levelNames' should be assigned as a cell array with the lowest level coming first and highest level coming last.
ModifyConfig('levelNames', {'Cluster', 'Group', 'Session', 'Site', 'Day', 'Days'});

levelAbbr' and 'NamePattern' also need to be assigned in order of the levels
ModifyConfig('levelAbbr', 'cgns');
ModifyConfig('NamePattern', {'cluster00s', 'group0000', 'session00', 'site00'})

For 'EqualName', the common-used name of certain level should be in the first place followed by '/' and the equivalent names. The equivalent names should be separated by '/' as well.
ModifyConfig('EqualName', {'Group/Sort/HighPass/Eye/EyeFilt/Lfp'});

For the way of calling *ModifyConfig.m*, you are strongly suggested to modify one component at one time. It prevents typing long command line and making typo errors.

Chapter 4 Create Child Class Object

DPV Toolbox provides a child class template @dirfiles. You may simply copy this class, paste it in the proper directory, and rename the class as well as the constructor function. There are two examples in the toolbox, @checkFiles and @plusFiles.

4.1 Contents in Child Class Template

The class @dirfiles has the following functions.

- Constructor – used to create an object with the appropriate input arguments passed. The constructor has the same name as the class. It is able to handle three possible combinations of input arguments:
    - ❖ No input argument – the constructor should create a default object

    - ❖ Object input argument – if the first argument in the argument list is an object of the same class, the constructor simply returns the object
    - ❖ Data input argument – the constructor creates the object using the input data
- *get.m* – used to access class properties
- *plus.m* – used to combine two objects in the same class to get the combined object
- *plot.m* – used to create plot using input object and input argument for visualizing analysis results
- *subsasgn.m* and *subsref.m* – used to define the behavior of indexing for the class

Usually, only the constructor and *plot.m* need to be modified for user-defined object.

4.2 Example classes

There are two tutorial classes in the toolbox, @checkFiles and @plusFiles. @checkFiles tests single analysis, and checks how much file in each directory for total number, '.mat' files, '.bin' files and '.txt' files. @plusFiles tests combinational analysis, and checks the sum of the files in specific directories the same as @ checkFiles. The constructors of the two test classes are created strictly by modifying the template of child class constructor, and the plot,m functions are created by modifying the template of plot.m in @dirfiles.

4.3 Modification in Constructor

There are five ways in total for template constructors to obtain the objects. Then either *createEmptyObject.m* or *createObject.m* is called.

a) Create empty object using arguments only;
b) No input arguments, create empty object;
c) If the first argument passed is just an object of the same class, return the very object;
d) If the previously saved object has the same requested arguments, load the previously saved object;
e) If none of a), b), c) and d) is satisfied, create object according to the input arguments.

In the fourth way, the program will do argument checking and comparison. However, some input arguments do not affect object creation, like 'RedoLevels' and 'SaveLevels'. These arguments must be neglected during argument checking. You need to specify them at the beginning of the constructor.

Example:
Args.UnimportantArgs = {'RedoLevels','SaveLevels'};

The main part for modification is the two functions *createEmptyObject.m* and *createObject.m*. They need to be developed carefully for object creation. If the object creation requires additional calculation before calling the two functions, the calculation should be added to where indicated by "IMPORTANT NOTICE!" in the template constructor.

4.4. Modification of plot.m

This function provides two types of plots, individual plot and population plot. Individual plot plots one data set at a time, while population plot plots all data together. You may see the codes in the template *plot.m*. Please add code in where indicated by "add code for plot options here".

Chapter 5 Data Processing

Two process functions are provided in DPV Toolbox, *ProcessLevel.m* and *ProcessDirs.m*.

5.1 ProcessLevel.m

NPTDATA/PROCESSLEVEL
NPTDATA/PROCESSCOMBINATION

Checks the local directory for data to process and returns the processed object, including individual analysis and combinational analysis. For the latter, used together with *ProcessCombination.m*.

Snytax
[robj,data] = ProcessLevel(obj,varargin);
[robj,data] = ProcessCombination(obj,varargin);

Optional Input arguments list of ProcessLevel.m

Input Argument Name	Explanations
Levels	Specifies the name of the highest processed level.
Include	Process selected items instead of all items found in the local directory, followed by a directory name or a cell array containing a list of directories.
Exclude	Skips the directories or items specified, followed by same items as in 'Include' . 'Include' and 'Exclude' cannot be specified together.
LevelObject	Indicates that the object should be instantiated at a specific level
AnalysisLevel	Specifies the AnalysisLevel property of the object Accept value: 'Single', 'All', 'Pairs', 'AllIntra<Level>', 'AllPairs'.

DataInit	Specifies the initial value of data.
nptLevelCmd	Specifies the level perform the following command of each directory in that level.
DataPlusCmd	Specified if only need to plus the data of the object.

Optional Input arguments list of ProcessCombination.m

Input Argument Name	Explanations	Accepted Values
Levels	Specifies the name of the highest processed level.	<Highest Level Name>
AnalysisRelation	Specifies the type of relationship of combination.	'Intra<Level>', 'Inter<Level>', '<Level>'(gives both of intra and inter)
AnalysisLevel	Specifies the AnalysisLevel property of the object.	'Single', 'All', 'Pairs', 'AllIntra<Level>', 'AllPairs'
DataInit	Specifies the initial value of data.	
nptComboCmd	Specifies the level perform the following command of each directory in that level.	
DataPlusCmd	Specified if only need to plus the data of the object.	

ProcessCombination.m inherits input argumrntes from *ProcessLevel.m*, if 'AnalysisLevel' is specified not 'Single', this function will automatically called by *ProcessLevel.m*. You only need to specify the argument 'AnalysisRelation' for it, and do not call it manually.

```
AnalysisRelation: Specifies the relationship of cpmbination.
        Accept values are:
                'Intra<Level>': give intra-level combination analysis
                'Inter<Level>': give inter-level combination analysis
                '<Level>': give both intra-level and inter-level analysis
        e.g.: 'AnalysisRelation','IntraSession'
              'AnalysisRelation','InterHour'

 AnalysisLevel:  Specifies the AnalysisLevel property of OBJ.
        Accepted values are:
                'All'            Groups of all intra-level directories for all <Level>s (a)
                'Pairs'          Pairs of directories.
                'AllIntra<Level>'   Groups of intra-level directories within one <Level>.
                'AllPairs'       Groups of intra-level pairs.(p)
```

Description and example codes:

Example data set levels are (lowest to highest):
Second, Minute, Hour, Hours

a = ProcessLevel(checkFiles, 'Levels', 'Hours'); processes the entire data set, instantiate the object in the lowest level, and returns a checkFiles object.
a = ProcessLevel(checkFiles, 'Levels', 'Hours', 'LevelObject','Minute'); processes the entire data set, instantiate the object in the level Minute, and returns a checkFiles object.
a = ProcessLevel(checkFiles, 'Levels', 'Hours', 'Include','hour0001'); processes only the directory hour0001, instantiate the object in the lowest level, and returns a checkFiles object.

a = ProcessLevel(checkFiles, 'Levels', 'Hours', 'Include', selectCells); processes only the directories listed in the cell array selectedCells.

a = ProcessLevel(plusFiles, 'Levels', 'Hours', 'LevelObject','Second',… 'AnalysisRelation', 'InterMinute','AnalysisLevel', 'Pairs'); checks the local directory for directories indicated by 'AnalysisRelation' (Minute directory in example) and loops over them to find directories indicated by 'LevelObject' (Second directory in example), returns plusFiles object created for inter-minute pairs. Combinations of these Second directories are created in which other objects may be saved. The 'AnalysisLevel' property of OBJ determines which combinations are created.

a = ProcessLevel(plusFiles, 'Levels', 'Hours', 'LevelObject','Second',… 'AnalysisRelation', 'IntraMinute','AnalysisLevel', 'AllIntraMinute'); returns plusFiles object created for grouping of Second directories of same Minute.

a = ProcessLevel(plusFiles, 'Levels', 'Hours', 'LevelObject','Second',… 'AnalysisRelation', 'IntraMinute','AnalysisLevel', 'AllPairs'); returns plusFiles object created for grouping of intra-minute pairs of same Minute.

a = ProcessLevel(plusFiles, 'Levels', 'Hours', 'LevelObject','Second',… 'AnalysisRelation', 'IntraMinute','AnalysisLevel', 'All'); returns plusFiles object created for grouping of all Second directories.

5.2 ProcessDirs.m

NPTDATA/PROCESSDIRS

Process directories in parent object, instead of processing level by level.

Snytax
[OBJ,OBJ2] = ProcessDirs(OBJ,'Object',OBJECTNAME);

Description and example codes:

npta = ProcessLevel(nptdata,'Levels','Hours');
[npta, b] = ProcessDirs(npta,'Object','checkFiles');
npta is parent object contains desired directories. It is passes to *ProcessDirs.m*, which processes the directories and returns checkFiles object in b. You may specify the object constructor options during calling *ProcessDirs.m*. These are equivalent to the following command:
b = ProcessLevel(checkFiles, ‘Levels’,’Hours’);

```
npta = ProcessLevel(nptdata,'Levels','Hours',…
        'AnalysisRelation','Minute','AnalysisLevel','Pairs');
[npta, b] = ProcessDirs(npta,'Object','plusFiles');
```
These are equivalent to the following command:
```
b = ProcessLevel(plusFiles,'Levels','Hours',…
        'AnalysisRelation','Minute','AnalysisLevel','Pairs');
```

Chapter 6 GUI

Graphical User Interface (GUI) is another important feature for DPV Toolbox. The functions consist of GUI system in the toolbox is shown in Figure 1.

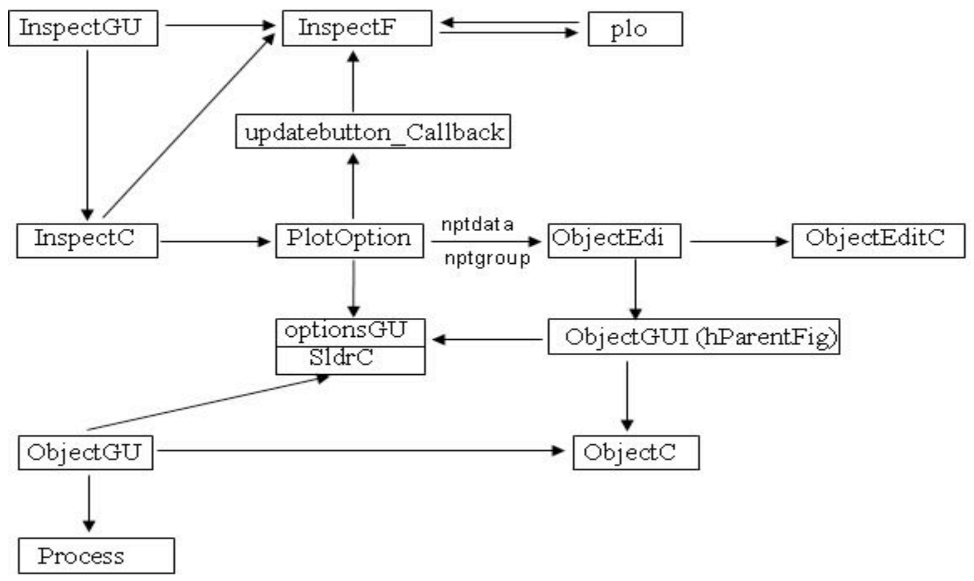


Figure 1: Functions Consist of GUI System

6.1 InspectGUI.m

NPTDATA/INSPECTGUI

Inspects objects, displays a graphical user interface to step through data contained in parent object, and is able to provide individual plots and population plots.

Syntax
FIG = InspectGUI(OBJ, VARARGIN)

Optional Input Argument List

Input Argument Name	Explanations
holdAxis	Specifies whether to hold the axis limits constant across plots.
multObjs	Specifies all the objects that should be plotted, should be assigned as a cell array.
addObjs	Specifies additional objects that are plotted at the mean time, and should be assigned as a cell array.
ObjectList	Specifies that the following cell array contains all the objects, together with the optional input arguments if any, that should be plotted.
optArgs	Specifies optional input arguments of plot options for the various objects.
OverPlot	Flag specifying that the plots for the different objects should be plotted on top of one another.

Description and example codes:

```
b = ProcessLevel(plusFiles,'Levels','Hours',
        'AnalysisRelation','Minute','AnalysisLevel','Pairs','Option','bin');
c = ProcessLevel(plusFiles,'Levels','Hours',…
        'AnalysisRelation','Minute','AnalysisLevel','Pairs','Option','mat');
InspectGUI(b,'addObjs', {c},'optArgs',{{'BinFile'},{'MatFile'}})
```
The plot is dhown in Figure 2.
```
InspectGUI(b,'addObjs', {c},'optArgs',{{'BinFile'},{'MatFile'}}, 'SP', [2,1])
```
The Plot is show in Figure 3.

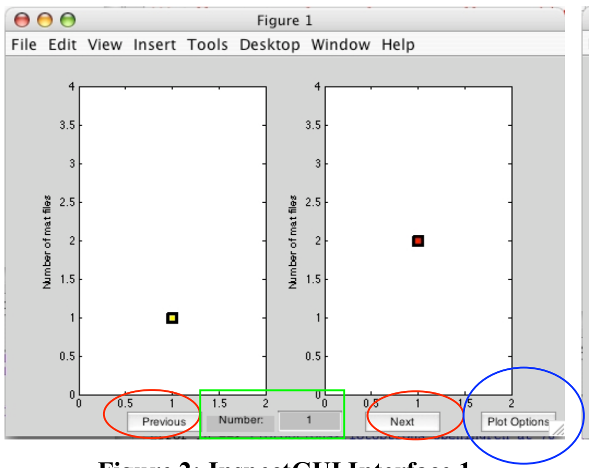


Figure 2: InspectGUI Interface 1

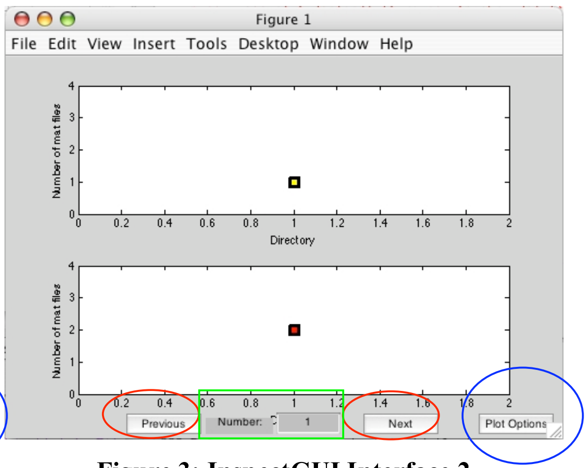
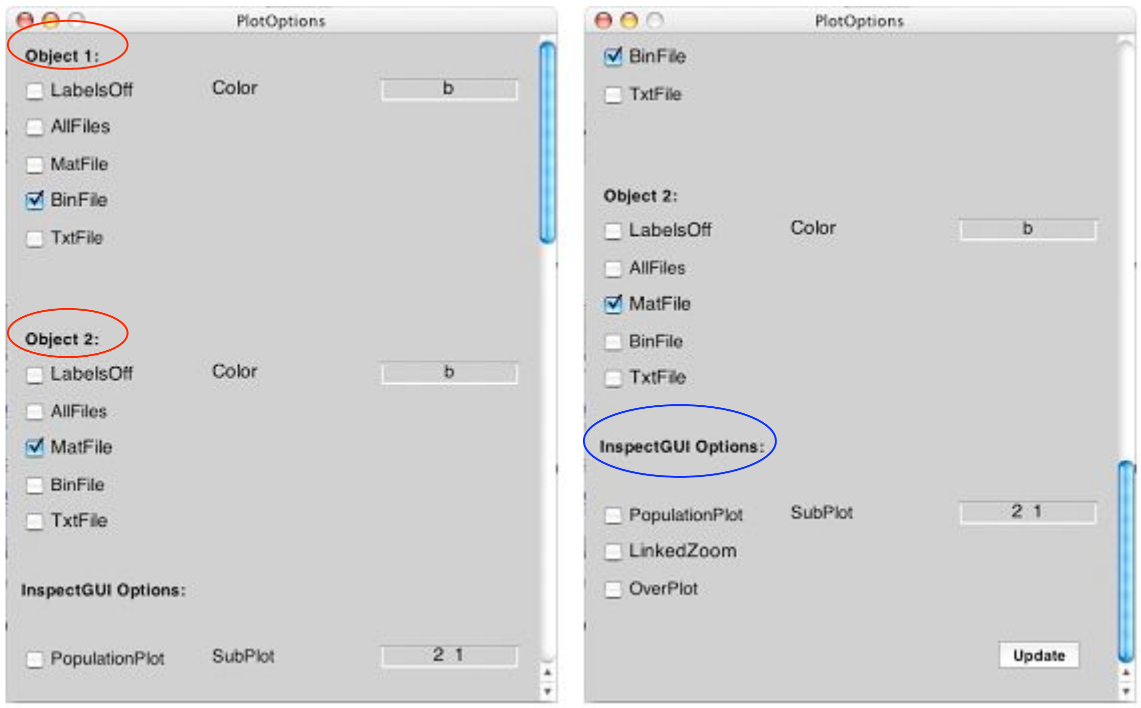


Figure 3: InspectGUI Interface 2

Both of The figures are the individual plots, each has two plots. There are three buttons and one edit text at the bottom of the figure.

- 'Previous' (indicated in red circles): changes all the plots to the previous data set;
- 'Next' (indicated in red circle): changes all the plots to the next data set;
- 'Number' (indicated in green rectangles): changes all the plots to the desired data set;
- 'Plot Options' (indicated in blue circles): displays PlotOptions GUI for the InspectGUI.

You may click the buttons 'Previous' and 'Next' to view different data sets, or change 'Number' to any data sets. If you click the 'Plot Option', the following will appear.



Object 1:
☐ LabelsOff   Color   b
☐ AllFiles
☐ MatFile
☑ BinFile
☐ TxtFile

Object 2:
☐ LabelsOff   Color   b
☐ AllFiles
☑ MatFile
☐ BinFile
☐ TxtFile

InspectGUI Options:
☐ PopulationPlot   SubPlot   2 1
☐ LinkedZoom
☐ OverPlot

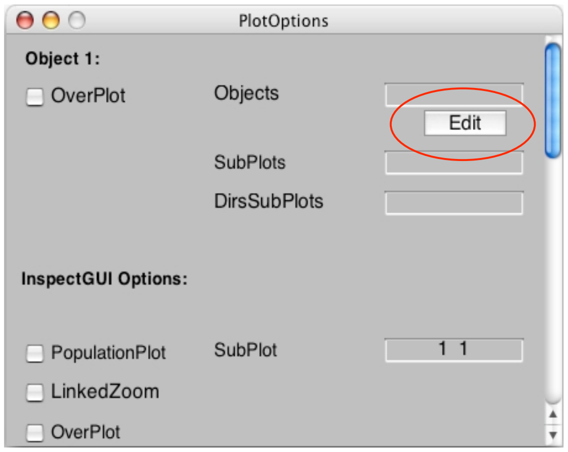
Figure 4: PlotOptions GUI

Figure 4 shows the same PlotOptions GUI with scrolling bar in different position. The two plot-option sets (indicated by red circles) belong to the two objects respectively, and the InspectGUI options (indicated by blue circle) apply to both objects. You may modify the options and click 'Update'; InspectGUI will update the plots with the new options.

InspectGUI.m also enables you to inspect plots of objects without creating them first, provided a parent object is passed to it.

```
npta = ProcessLevel(nptdata,'Levels','Hours','AnalysisRelation',…
        'Minute','AnalysisLevel', 'Pairs');
InspectGUI(npta,'Object',{'plusFiles'})
```

The InspectGUI interface will appear. Click 'Plot Options' to get PlotOptions GUI as shown in Figure 5, and click 'Edit' (indicated in red circle) to modify the object options. The ObjectEdit GUI will appear as shown in Figure 6.



SubPlot 1 1

Figure 5: Plot Options GUI 2

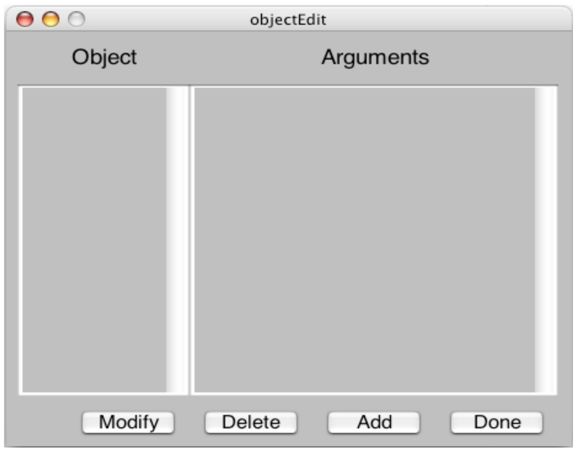
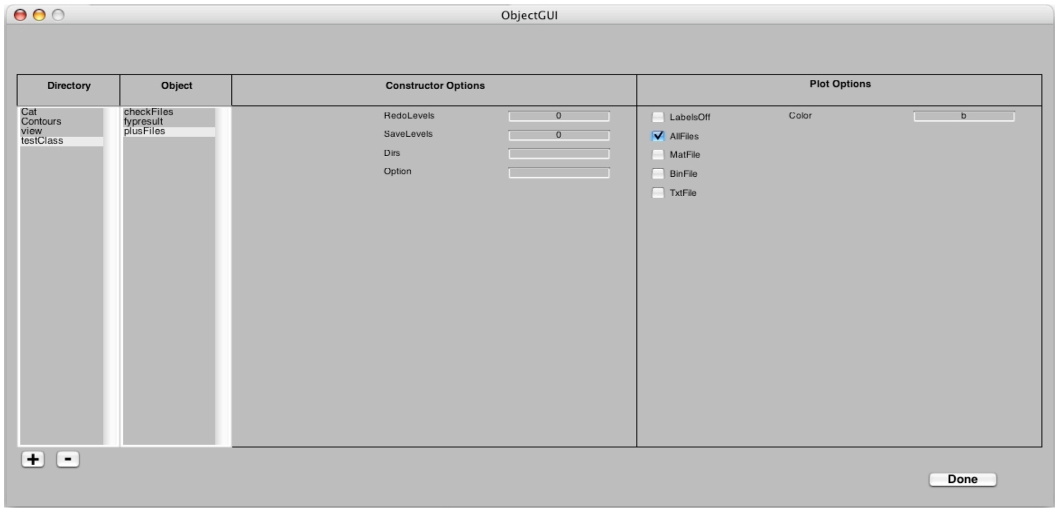


Figure 6: ObjectEdit GUI

Four buttons are at the bottom of ObjectEdit GUI.

- Modify: creates the corresponding ObjectGUI GUI (introduced later) with the selected object name and respective options to modify.
- Delete: deletes the selected object and remove the respective options.
- Add: creates an ObjectGUI GUI to add a new object with the respective options.
- Done: returns the object name(s) with the respective options in the form of string to PlotOptions GUI and replace the previous string in the editable text following 'Object' or 'Objects'.

You may click 'Modify' to modify constructor options and plot options of the existing object. If the object wanted does not exist, click 'Add' to create one. If either of the two is clicked, the ObjectGUI interface will appear as shown in Figure 7.



Directory	Object
Cat	checkFiles
Contours	fypresult
view	plusFiles
testClass	

RedoLevels 0
SaveLevels 0
Dirs
Option

LabelsOff
[✓ AllFiles]
MatFile
BinFile
TxtFile
Color b

Figure 7: ObjectGUI GUI

In the ObjectGUI interface, the first column is the directory list. The '+' and '-' under the list-box is for you to add and delete directories. Selecting a directory, the second column will display the respective object list. Selecting an object, the third column will display the respective constructor options, as well as the constructor options of the parent class if any, and the last column will display the plot options of the object. You may modify the options and click 'Done', and then the ObjectGUI is closed and ObjectEdit GUI is updated with the modified options. Click 'Done' in ObjectEdit GUI, it will be closed, and the Plot Options GUI is updated. Eventually, you will get the plot of the object using GUI environment.

6.2 ObjectGUI

Provides user to create objects in GUI environment, prevents typing long command line in Command Window.

Syntax
ObjectGUI(VARARGIN)
ObjectGUI

Description and example codes:

ObjectGUI.m has the following properties:

- Provides user to add and delete the directories, to select the directories of the object(s), as well as to modify the constructor options of the respective object.
- The list of directories, along with the respective object lists, is saved in the preference directory, where the configuration file is saved. The directory is returned by 'prefdir'.
- Can be called either with or without input arguments.

ObjectGUI.m can be called by other functions or GUIs with input arguments. In this way of calling, ObjectGUI provides the modification of the plot options of objects, as shown and explained in Figure 7. The first input argument is the handle of the calling figure. The second input argument is the handle of the object which contains some object names on the ObjectEdit GUI when the ObjectGUI interface is called to modify the specified object.

Figure 8 shows the ObjectGUI interface when *ObjectGUI.m* is called without any input arguments. It is slightly different from the one shown in Figure 7.

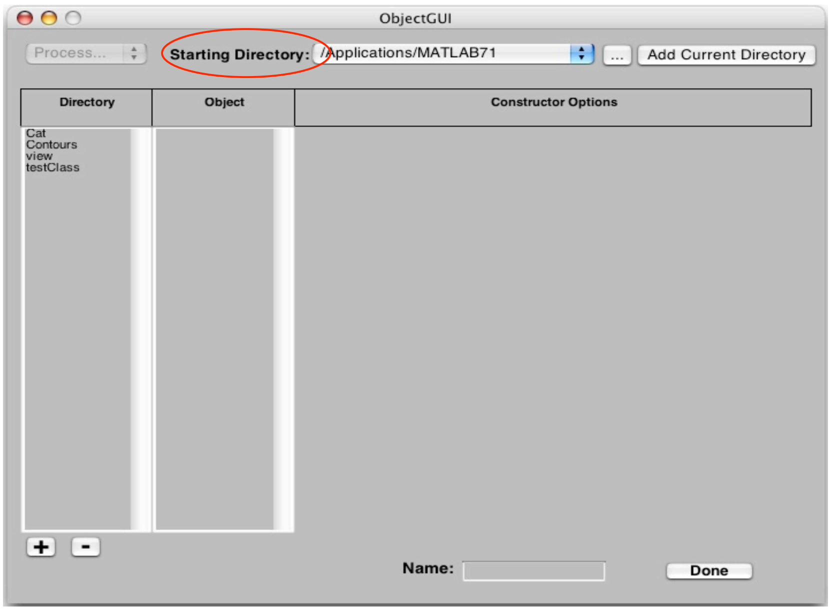


Figure 8: ObjectGUI GUI Called without Input Arguments

You may select the directory from the 'Starting Directory' (indicated in red circle) list or browse to a new directory by click the button '…' besides 'Starting Directory'. Once you select a directory, the 'Current Directory' in Matlab Command Window will change to the same one. If you change the directory in Matlab Command Window, please click the button 'Add Current Directory' in the upper right-hand corner to get the same directory.

The first column is the directory list. The '+' and '-' under the list-box is for you to add and delete directories. Selecting a directory, the second column will display the respective object list. Selecting an object, the third column will display the respective constructor options, as well as the constructor options of the parent class if any. The list box 'Process…' in the upper left-hand corner is activated for you to select either 'ProcessDirs' or 'ProcessLevel' (shown in Figure 9).

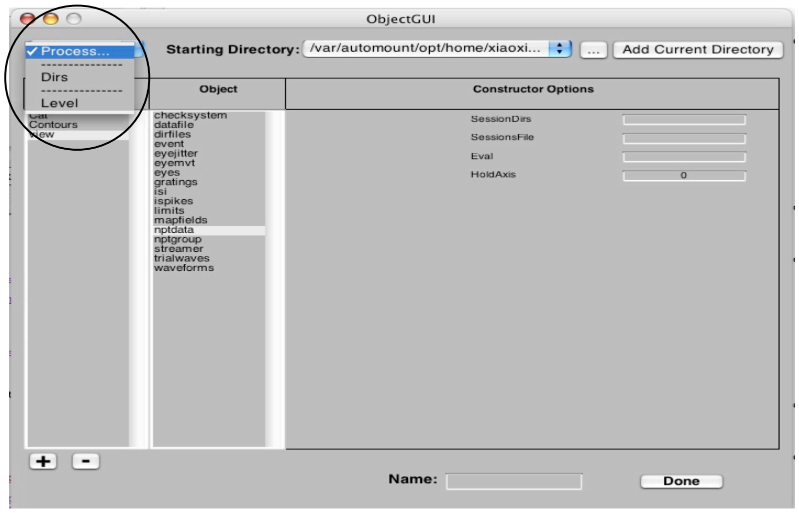
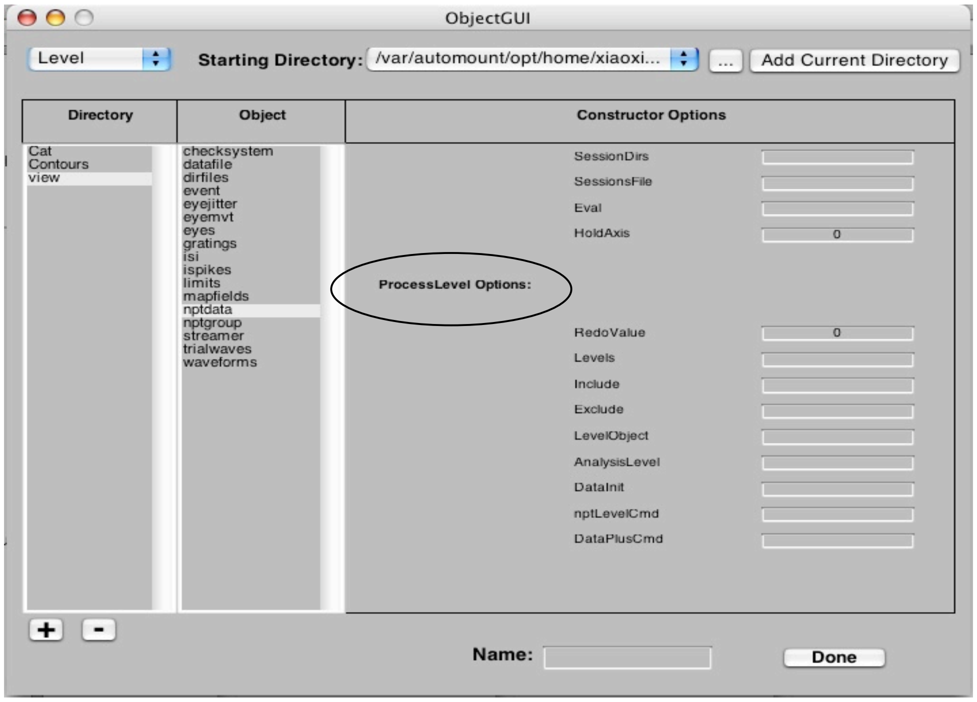


Figure 9: Select 'Process…' List Box

Once selection is done, the respective process options appear under the column of construction options. Figure 10 shows the ProcessLevel Options. If you specify the input argument ‘AnalysisLevel’ not to be Single, press ‘Enter’, the ProcessCombination Options will appear (shown in Figure 11).



Starting Directory: /var/automount/opt/home/xiaoxi...

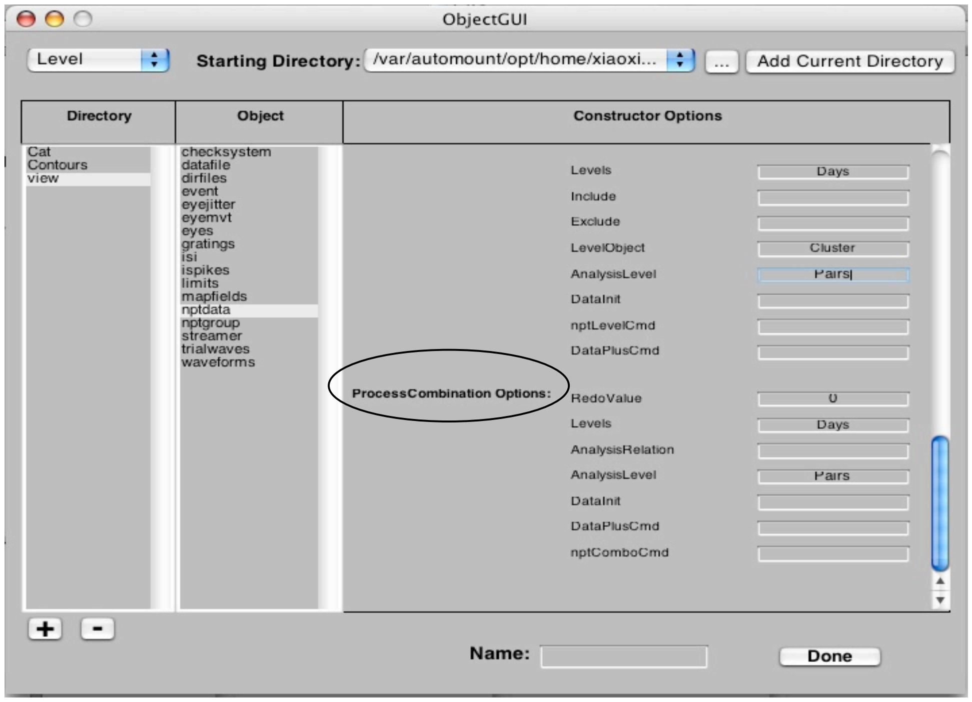
SessionDirs	
SessionsFile	
Eval	
HoldAxis	0

ProcessLevel Options:

RedoValue	0
Levels	
Include	
Exclude	
LevelObject	
AnalysisLevel	
DataInit	
nptLevelCmd	
DataPlusCmd	

Name:

Figure 10: ProcessLevel Options



Starting Directory: /var/automount/opt/home/xiaoxi...

Levels	Days
Include	
Exclude	
LevelObject	Cluster
AnalysisLevel	Pairs
DataInit	
nptLevelCmd	
DataPlusCmd	

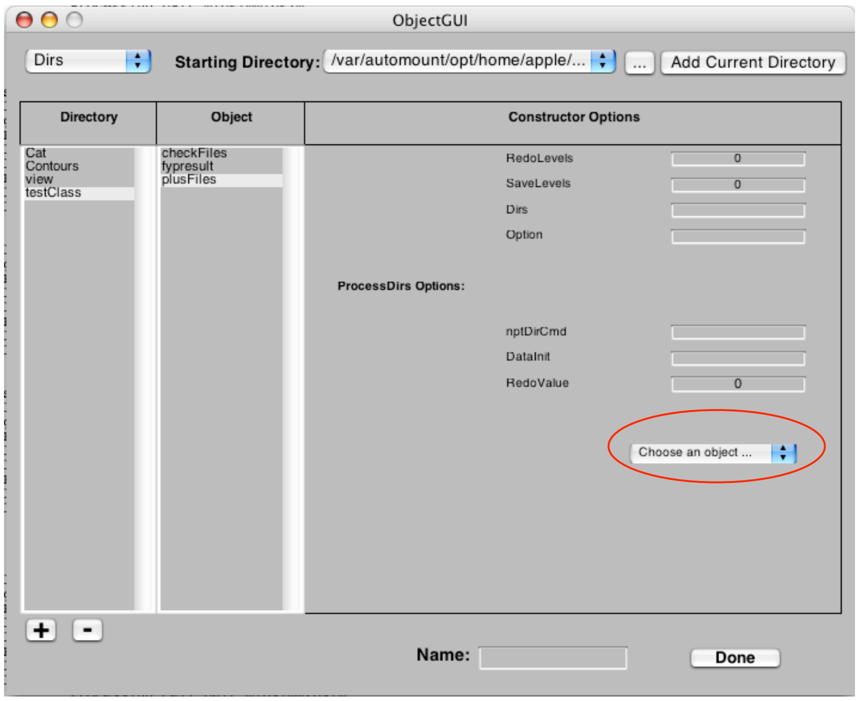
ProcessCombination Options:

RedoValue	0
Levels	Days
AnalysisRelation	
AnalysisLevel	Pairs
DataInit	
DataPlusCmd	
nptComboCmd	

Name:

Figure 11: ProcessCombination Options

Figure 12 shows the ProcessDirs Options. There is a list at the end of the option list (indicated in red circle). It provides you to choose one of the parent objects in the base workspace (as shown in Figure 13). You need to select an object.



Starting Directory: /var/automount/opt/home/apple/...

Directory: Cat, Contours, view, testClass

Object: checkFiles, fypresult, plusFiles

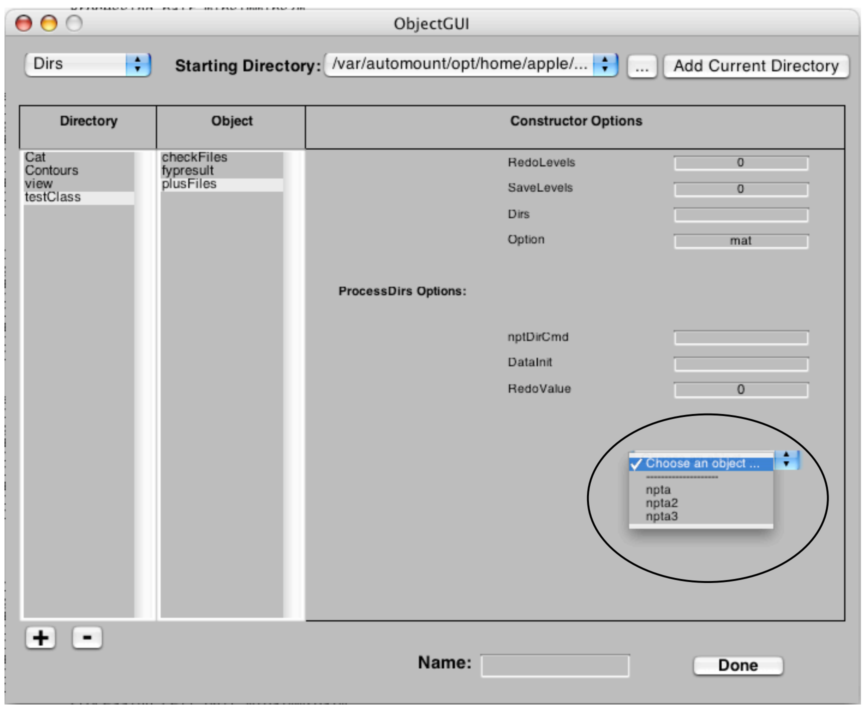
RedoLevels	0
SaveLevels	0
Dirs	
Option	

ProcessDirs Options:

nptDirCmd	
DataInit	
RedoValue	0
	Choose an object ...

Name:

Figure 12: ProcessDirs Options



Starting Directory: /var/automount/opt/home/apple/...

Directory: Cat, Contours, view, testClass

Object: checkFiles, fypresult, plusFiles

RedoLevels	0
SaveLevels	0
Dirs	
Option	mat

ProcessDirs Options:

nptDirCmd	
DataInit	
RedoValue	0

✓ Choose an object ...
----
npta
npta2
npta3

Name:

Figure 13: Objects in Base Workspace

Once you finish modifying the options, please give a name to the object in the editable text box following ‘Name’, and then click ‘Done’ to get the object created in the base workspace.